DOCKET NO.: IVGP-0002 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: **Hiang-Swee Chiang**	Confirmation No.: **2184**
Serial No.: **09/810,716**	Group Art Unit: 2193
Filing Date: **March 16, 2001**	Examiner: **William H. Wood**

For: **WEB APPLICATION GENERATOR**

Mail Stop Appeal-Brief Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDED APPEAL BRIEF PURSUANT TO 37 C.F.R. § 41.37

This brief is filed in support of Appellant's appeal from the rejections of claims 2, 4-29, 31-50, 52-64, 66-78, and 162-169. A Notice of Appeal was submitted on June 30, 2006.

An initial appeal brief was submitted January 31, 2007. This brief is filed in response to a Notice of Non-Compliant Appeal Brief mailed on April 3, 2007.

Table of Contents

This brief is being filed in support of Appellant's appeal from the rejections of claims 2, 4-29, 31-50, 52-64, 66-78 and 162-169 dated May 17, 2006. A Notice of Appeal was filed on June 30, 2006.

I. REAL PARTY IN INTEREST

Guttenberg Printing LLC is the real party of interest by virtue of an assignment recorded October 4, 2004, at Reel 015213, Frame 0050.

II. RELATED APPEALS AND INTERFERENCES

The Applicant, the Applicant's legal representative and the real party-in-interest are unaware of any appeals or interferences that are related to this appeal.

III. STATUS OF CLAIMS

Claim 1	Cancelled
Claims 2	Rejected and On Appeal
Claim 3	Cancelled
Claims 4-29	Rejected and On Appeal
Claim 30	Cancelled
Claims 31-50	Rejected and On Appeal
Claim 51	Cancelled
Claims 52-64	Rejected and On Appeal
Claim 65	Cancelled
Claims 66-78	Rejected and On Appeal
Claim 79-161	Cancelled
Claims 162-169	Rejected and On Appeal
Claims 170-174	Cancelled

IV. STATUS OF AMENDMENTS

The Appellant's amendment of February 21, 2006 has been entered. There have been no further amendments since the issuance of the final office action dated May 17, 2006.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Applicants have disclosed methods and systems for generating a web application. In an exemplary method, a web application server receives input files from graphic designers or business analysts. The input files may comprise at least one web application graphical user interface. The web application server determines if an application framework code is available for the web application, and retrieves the application framework code from an application directory. If the application framework code is not available for the web application, then the web application server generates the application framework code, along with a business logic foundation code, an event handler skeleton and a graphical user interface code. Generating the event handler skeleton comprises parsing the input files, reviewing each parsed input file for a tag type, attribute name and attribute value, and determining an event handler method based on the tag type, attribute name and attribute value.

The web application server receives web application business logic objects and event handlers from the web developers, and organizes the application framework code, web application business logic objects and event handler methods into web application source code. The web application server then compiles the web application source code. Modified input files may subsequently be received by the web application server from the graphic designers or business analysts. The modified input files are compiled and dynamically bound with the compiled web application source code at runtime.

There are ten (10) independent claims: 2; 27; 48; 64; 78; 162; 163; 164; 165; and 166. In accordance with 37 C.F.R. § 41.37(c)(v), provided below for each of the independent claims is a concise explanation of the defined subject matter including references to specification and characters.

A. Independent Claim 2

Claim 2 is directed to a method of generating computer code for a web application (Figs. 5, 6, and 7; page 13, ln. 9 – page 22, ln. 20). In the recited method, input files are received with at least one of the input files comprising a web application graphical user interface. (Fig. 5, reference 510; Fig. 6; page 13, ll. 9-18). An application framework code and event handler skeleton are then generated. (Fig. 7, reference 710; page 13, ln. 19 – page 19, ln. 9).

Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2). The web application source code is then bound with the input files at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20).

B. Independent Claim 27

Claim 27 is directed to a method of generating computer code for a web application. (Figs. 5, 6, and 7, p. 13, ln 9 – p. 22, ln 20). In the recited method, input files are received, with the input files comprising at least one web application graphical user interface. (Fig. 5, reference 510; Fig. 6; p. 13, ll. 9-18). Then an application framework code is retrieved from an application directory. (Fig. 7, reference 710; p. 13, ln. 19 – p. 14, ln. 18).

An event handler skeleton is also generated. (p. 18, ln. 3 – p. 19, ln. 9). Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2). The web application source code is then bound with the input files at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20).

C. Independent Claim 48

Claim 48 is directed to a method of generating computer code for a web application (Figs. 5, 6, and 7, p. 13, ln 9 – p. 22, ln 20). In the recited method, input files are

received, with the input files comprising at least one web application graphical user interface. (Fig. 5, reference 510; Fig. 6; p. 13, ll. 9-18). An application framework code and an event handler skeleton are then generated. (Fig. 7, reference 710; page 13, ln. 19 – page 19, ln. 9).

Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2).

Modified input files are then received. (p. 20, ln. 12 – p. 21, ln. 5). The modified input files are then bound with the web application source code at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20; p. 20, ln. 12 – p. 21, ln. 5).

D. Independent Claim 64

Claim 64 is directed to a method of generating computer code for a web application. (Figs. 5, 6, and 7, p. 13, ln 9 – p. 22, ln 20). In the recited method, input files are received, with the input files comprising at least one web application graphical user interface. (Fig. 5, reference 510; Fig. 6; p. 13, ll. 9-18). Then an application framework code is retrieved from an application directory. (Fig. 7, reference 710; p. 13, ln. 19 – p. 14, ln. 18).

An event handler skeleton is also generated. (p. 18, ln. 3 – p. 19, ln. 9). Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p.

22, ln. 2). Modified input files are then bound with the web application source code at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20; p. 20, ln. 12 – p. 21, ln. 5).

E. Independent Claim 78

Claim 78 is directed to a method of generating computer code for a web application. (Figs. 5, 6, and 7, p. 13, ln 9 – p. 22, ln 20). In the recited method, an event handler skeleton is generated. (p. 18, ln. 3 – p. 19, ln. 9). Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on the one or more tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, business logic foundation code, event handler skeleton, and a graphical user interface code are received. (Fig. 5, reference 520; p. 13, ll. 9-18; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, web application business logic objects are prepared based on the business logic foundation code. (Fig. 5, reference 535; p. 19, ll. 4-9; p. 22, ll. 4-20).

F. Independent Claim 162

Claim 162 is directed to a device (Figs. 1, 2, and 3, reference 100; p. 8, ln, 1 – p. 10, ln. 19) for generating computer code for a web application. The device comprises a processor (Fig. 3, reference 305; p. 9, ln. 20 – p. 10, ln. 11) and a storage device (Fig. 3, reference 325; p. 11, ln. 3-10) for storing a program for controlling the processor (Fig. 3, reference 330; p. 11, ln. 11-13).

The processor with the program is operative to perform a method. The method comprises generating a business logic foundation code, an event handler skeleton and a graphical user interface code. (Fig. 6; p. 13, ll. 9 – 19).

Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

The method further comprises receiving web application business logic objects and event handler methods from a web developer. (Fig. 6; p. 18, ln. 3 – p. 20, ln. 2)

The method still further comprises organizing the application framework code, the web application business logic objects and the event handler methods into web application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2). The web application source code is then compiled. (Fig. 5, reference 535; p. 22, ll. 4-20). Modified input files are compiled at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20). The modified input files and compiled web application source code are bound at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20; p. 20, ln. 12 – p. 21, ln. 5).

G. Independent Claim 163

Claim 163 is directed to a device (Figs. 1, 2, and 3, reference 100; p. 8, ln, 1 – p. 10, ln. 19) for generating computer code for a web application. The device comprises a processor (Fig. 3, reference 305; p. 9, ln. 20 – p. 10, ln. 11) and a storage device (Fig. 3, reference 325; p. 11, ln. 3-10) for storing a program for controlling the processor (Fig. 3, reference 330; p. 11, ln. 11-13).

The processor with the program is operative to perform a method. In the recited method, input files are received with at least one of the input files comprising a web application graphical user interface. (Fig. 5, reference 510; Fig. 6; page 13, ll. 9-18). An application framework code and event handler skeleton are then generated. (Fig. 7, reference 710; page 13, ln. 19 – page 19, ln. 9).

Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2). The web application source code is then bound with the input files at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20).

H. Independent Claim 164

Claim 164 is directed to a device (Figs. 1, 2, and 3, reference 100; p. 8, ln, 1 – p. 10, ln. 19) for generating computer code for a web application. The device comprises a

processor (Fig. 3, reference 305; p. 9, ln. 20 – p. 10, ln. 11) and a storage device (Fig. 3, reference 325; p. 11, ln. 3-10) for storing a program for controlling the processor (Fig. 3, reference 330; p. 11, ln. 11-13).

The processor with the program is operative to perform a method. In the recited method, input files are received, with the input files comprising at least one web application graphical user interface. (Fig. 5, reference 510; Fig. 6; p. 13, ll. 9-18). Then an application framework code is retrieved from an application directory. (Fig. 7, reference 710; p. 13, ln. 19 – p. 14, ln. 18).

An event handler skeleton is also generated. (p. 18, ln. 3 – p. 19, ln. 9). Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2). The web application source code is then bound with the input files at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20).

I. Independent Claim 165

Claim 165 is directed to a device (Figs. 1, 2, and 3, reference 100; p. 8, ln, 1 – p. 10, ln. 19) for generating computer code for a web application. The device comprises a processor (Fig. 3, reference 305; p. 9, ln. 20 – p. 10, ln. 11) and a storage device (Fig. 3, reference 325; p. 11, ln. 3-10) for storing a program for controlling the processor (Fig. 3, reference 330; p. 11, ln. 11-13).

The processor with the program is operative to perform a method. In the recited method, input files are received, with the input files comprising at least one web application graphical user interface. (Fig. 5, reference 510; Fig. 6; p. 13, ll. 9-18). An application framework code and an event handler skeleton are then generated. (Fig. 7, reference 710; page 13, ln. 19 – page 19, ln. 9).

Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a

tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2).

Modified input files are then received. (p. 20, ln. 12 – p. 21, ln. 5). The modified input files are then bound with the web application source code at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20; p. 20, ln. 12 – p. 21, ln. 5).

J. Independent Claim 166

Claim 166 is directed to a device (Figs. 1, 2, and 3, reference 100; p. 8, ln, 1 – p. 10, ln. 19) for generating computer code for a web application. The device comprises a processor (Fig. 3, reference 305; p. 9, ln. 20 – p. 10, ln. 11) and a storage device (Fig. 3, reference 325; p. 11, ln. 3-10) for storing a program for controlling the processor (Fig. 3, reference 330; p. 11, ln. 11-13).

The processor with the program is operative to perform a method. In the recited method, input files are received, with the input files comprising at least one web application graphical user interface. (Fig. 5, reference 510; Fig. 6; p. 13, ll. 9-18). Then an application framework code is retrieved from an application directory. (Fig. 7, reference 710; p. 13, ln. 19 – p. 14, ln. 18).

An event handler skeleton is also generated. (p. 18, ln. 3 – p. 19, ln. 9). Generating an event handler skeleton comprises parsing at least one input file (Fig. 7, reference 720; p. 14, ln. 19 – p. 15, ln. 1), reviewing the parsed input file for one or more of a tag type, an attribute name, and an attribute value (Fig. 7, reference 725; p. 14, ln. 21 – p. 16, ln. 4), and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value (Fig. 7, reference 730; p. 15, ln. 3 – p. 16, ln 14).

In the recited method, web application business logic objects and event handler methods are then received. (Fig. 5, reference 520; p. 20, ln. 3 – p. 21, ln. 14). Thereafter, the application framework code, the web application business logic objects, and the event handler methods are organized into application source code. (Fig. 5, reference 530; p. 21, ln. 15 – p. 22, ln. 2).

Modified input files are then received. (p. 20, ln. 12 – p. 21, ln. 5). The modified input files are then bound with the web application source code at runtime. (Fig. 5, reference 535; p. 22, ll. 4-20; p. 20, ln. 12 – p. 21, ln. 5).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The rejections of claims 2, 4-29, 31-50, 52-64, 66-78, and 162-169 as unpatentable under 35 U.S.C. §103(a) over Lau (U.S. Patent No. 5,987,247) in view of Lindhorst et al. (U.S. Patent No. 6,337,696) in further view of Quaeler-Bock et al.(U.S. Patent No. 6,023,271). This is the only grounds of rejection.

VII. ARGUMENT

Claims 2, 4-29, 31-50, 52-64, 66-78, and 162-169 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Lau in view of Lindhorst in further view of Quaeler-Bock. Applicant respectfully submits that the cited references fail to render the claims obvious.

A. Claims 2, 4-29, 31—50, 52-64, 66-78, and 162-169 Are Not Obvious

Representative independent claim 2 recites, amongst other language, "generating an event handler skeleton compris[ing] . . . determining an event handler method **based on one or more of the tag type, the attribute name and attribute value**." Applicant respectfully submits that the recited references do not teach this language and cannot possibly suggest the recited combination. *See* M.P.E.P. § 2143.03 ("[t]o establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art.").

The Examiner acknowledges that Lau does not teach "generating an event handler skeleton compris[ing] . . . parsing at least one input file; reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and determining an event handler method based on one or more of the tag type, the attribute name and the attribute value." (Final Office Action dated may 17, 2006, p. 3). Instead, the Examiner alleges that Lindhorst teaches this language. Applicant respectfully disagrees.

Lindhorst discloses a system for generating and editing event handlers that link events triggered on one object to actions taken on one or more different objects. (Abstract). The

system includes a user interface having an event pane, action pane, and code pane. (*Id.*) According to Lindhorst, **a user selects** an event icon in the event pane to link that event to a desired action in the action pane. (Abstract).

Thus, the purpose behind Lindhorst is to allow users to selectively link events to desired actions. In contrast, claim 2 recites "generating an event handler skeleton." Lindhorst does not even mention the concept of an *event handler skeleton*, and certainly does not teach "*generating* an event handler skeleton."

Furthermore, Lindhorst does not teach "wherein generating an event handler skeleton comprises . . . **determining an event handler method based on one or more of the tag type, the attribute name and attribute value**." The May 17, 2006 Office Action alleges that Lindhurst teaches this recited claim language at column 13, lines 22 through 64. (Office Action, pp. 3-4). But, in fact, the referenced portion of Lindhorst simply teaches that events are displayed on an event pane of a user interface (col. 13, ll. 22-24) while methods are displayed on an action pane of the user interface (col. 13, ll. 65-67). The referenced language of Lindhorst does not teach "determining an event handler method based on one or more of the tag type, the attribute name and attribute value." Indeed, Lindhorst teaches that a ***user selects*** an event icon in the event pane to link that event to a desired action in the action pane. (Abstract). There is no indication that the user makes a determination "**based on** one or more of the tag type, the attribute name and attribute value."

As noted in the May 17, 2006 Office Action (page 13), Table 2 in Lindhorst discloses a listing of scriptable HTML tags with events and methods. But Lindhorst does ***not*** indicate that the listed tags, events, or methods are employed by the user to "determine[e] an event handler method." Rather, as Lindhorst itself acknowledges, Table 2 merely provides a listing of exemplary scriptable tags as defined in the HTML 3.0 specification. (Col. 13, ll. 24-26). The listed tags are those that are defined in the HTML specification as being scriptable (as compared to those tags that are not scriptable). Thus, Table 2 of Lindhorst does not teach or suggest that the "user" described in Lindhorst makes a determination "**based on** one or more of the tag type, the attribute name and attribute value."

The remaining reference relied upon for the rejection, Quaeler-Bock, discloses a system for programming graphical user interfaces. (Abstract). But Quaeler-Bock also does not teach or suggest "wherein generating an event handler skeleton comprises . . . **determining an event handler method based on one or more of the tag type, the**

attribute name and attribute value." Indeed, the May 17, 2006 Office Action does not even suggest that this is the case.

Therefore, because neither Lau, Lindhorst, nor Quaeler-Bock teach all of the recited claim language, the cited references do not render obvious claim 2 and all claims depending therefrom. *See* M.P.E.P. § 2143.03. For similar reasons, independent claims 27, 48, 64, 78, 162, 163, 164, 165, and 166, and all claims depending therefrom are patentable over the cited references. Withdrawal of the rejections under 35 U.S.C. § 103(a) is respectfully requested.

VIII. CONCLUSION

Applicant respectfully submits that the rejection of claims under 35 U.S.C. § 103(a) was improper. For the all of the foregoing reasons, Applicant respectfully requests that the Board reverse the rejections.

Date: September 4, 2007

/John E. McGlynn/
John E. McGlynn
Registration No. 42,863

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439

IX. CLAIMS APPENDIX

1. (Canceled)

2. (Rejected and On Appeal) A method of generating computer code for a web application, comprising:

receiving input files, wherein the input files are at least one web application graphical user interface;

generating an application framework code and an event handler skeleton, wherein generating an event handler skeleton comprises:

parsing at least one input file;

reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and

determining an event handler method based on one or more of the tag type, the attribute name and the attribute value;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework code, the web application business logic objects and the event handler methods into application source code; and

binding the web application source code with the input files at runtime.

3. (Canceled)

4. (Rejected and On Appeal) The method of claim 2, wherein the web application source code is generated in an object-oriented programming language.

5. (Rejected and On Appeal) The method of claim 4, wherein the object-oriented programming language is Java.

6. (Rejected and On Appeal) The method of claim 4, wherein the object-oriented programming language is C++.

7. (Rejected and On Appeal) The method of claim 2, further comprising determining if the application framework code is available for the web application.

8. (Rejected and On Appeal) The method of claim 2, further comprising generating a business logic foundation code.

9. (Rejected and On Appeal) The method of claim 2, further comprising generating a graphical user interface code.

10. (Rejected and On Appeal) The method of claim 9, wherein generating a graphical user interface code is based on the input files.

11. (Rejected and On Appeal) The method of claim 2, wherein generating an event handler skeleton is based on the input files.

12. (Rejected and On Appeal) The method of claim 2, further comprising compiling the web application source code.

13. (Rejected and On Appeal) The method of claim 2, further comprising interpreting the web application source code.

14. (Rejected and On Appeal) The method of claim 2, wherein the input files are in XML format.

15. (Rejected and On Appeal) The method of claim 2, wherein the input files are in HTML format.

16. (Rejected and On Appeal) The method of claim 2, wherein the input files are in cHTML format.

17. (Rejected and On Appeal) The method of claim 2, wherein the input files are in WML format.

18. (Rejected and On Appeal) The method of claim 2, further comprising receiving modified input files.

19. (Rejected and On Appeal) The method of claim 18, further comprising compiling the modified input files at runtime.

20. (Rejected and On Appeal) The method of claim 19, further comprising binding the web application source code with the compiled modified input files at runtime.

21. (Rejected and On Appeal) The method of claim 20, wherein the modified input files are compiled into DOM objects at runtime.

22. (Rejected and On Appeal) The method of claim 18, further comprising interpreting the modified input files at runtime.

23. (Rejected and On Appeal) The method of claim 22, further comprising binding the web application source code with the interpreted modified input files at runtime.

24. (Rejected and On Appeal) The method of claim 2, further comprising generating application runtime properties.

25. (Rejected and On Appeal) The method of claim 2, further comprising generating application SQL statements.

26. (Rejected and On Appeal) The method of claim 2, wherein the application framework code comprises an application object and a sen/let web application framework object.

27. (Rejected and On Appeal) A method of generating computer code for a web application, comprising:

receiving input files, wherein the input files are at least one web application graphical user interface;

retrieving an application framework code from an application directory;

generating an event handler skeleton;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework, code, the web application, business logic objects and the event handler methods into application source code; and

binding the web application source code with the input files at runtime_and wherein generating an event handler skeleton further comprises:

parsing at least one input file;

reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and determining an event handler method based on the tag type, the attribute name and the attribute value.

28. (Rejected and On Appeal) The method of claim 27, further comprising retrieving a business logic foundation code.

29. (Rejected and On Appeal) The method of claim 27, further comprising generating a business logic foundation code.

30. (Canceled)

31. (Rejected and On Appeal) The method of claim 27, wherein the web application source code is generated in an object-oriented programming language.

32. (Rejected and On Appeal) The method of claim 27, further comprising determining if the application framework code is available for the web application.

33. (Rejected and On Appeal) The method of claim 27, further comprising generating a graphical user interface code.

34. (Rejected and On Appeal) The method of claim 33, wherein generating a graphical user interface code is based on the input files.

35. (Rejected and On Appeal) The method of claim 27, wherein generating an event handler skeleton is based on the input files.

36. (Rejected and On Appeal) The method of claim 27, further comprising compiling the web application source code.

37. (Rejected and On Appeal) The method of claim 27, further comprising interpreting the web application source code.

38. (Rejected and On Appeal) The method of claim 27, wherein the input files are in XML format.

39. (Rejected and On Appeal) The method of claim 27, wherein the input files are in HTML format.

40. (Rejected and On Appeal) The method of claim 27, wherein the input files are in cHTML format.

41. (Rejected and On Appeal) The method of claim 27, wherein the input files are in WML format.

42. (Rejected and On Appeal) The method of claim 27, further comprising receiving modified input files.

43. (Rejected and On Appeal) The method of claim 42, further comprising compiling the modified input files at runtime.

44. (Rejected and On Appeal) The method of claim 43, further comprising binding the web application source code with the compiled modified input files at runtime.

45. (Rejected and On Appeal) The method of claim 42, further comprising interpreting the modified input files at runtime.

46. (Rejected and On Appeal) The method of claim 45, further comprising binding the web application source code with the interpreted modified input files at runtime.

47. (Rejected and On Appeal) The method of claim 27, wherein the application framework code comprises an application object and a servlet web application framework object.

48. (Rejected and On Appeal) A method of generating computer code for a web application, comprising:

receiving input files, wherein the input files are at least one web application graphical user interface;

generating an application framework code and an event handler skeleton;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework code, the web application business logic objects and the event handler methods into web application source code;

receiving modified input files; and

binding the modified input files with the web application source code at runtime and wherein generating an event handler skeleton further comprises:

parsing at least one input file;

reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value; and

determining an event handler method based on the one or more of tag type, the attribute name and the attribute value.

49. (Rejected and On Appeal) The method of claim 48, further comprising compiling the modified input files at runtime.

50. (Rejected and On Appeal) The method of claim 48, further comprising interpreting the modified input files at runtime.

51. (Canceled)

52. (Rejected and On Appeal) The method of claim 48, wherein the web application source code is generated in an object-oriented programming language.

53. (Rejected and On Appeal) The method of claim 48, further comprising determining if the application framework code is available for the web application.

54. (Rejected and On Appeal) The method of claim 48, further comprising generating a business logic foundation code.

55. (Rejected and On Appeal) The method of claim 48, further comprising generating a graphical user interface code.

56. (Rejected and On Appeal) The method of claim 48, further comprising compiling the web application source code.

57. (Rejected and On Appeal) The method of claim 48, further comprising interpreting the web application source code.

58. (Rejected and On Appeal) The method of claim 48, wherein the input files are in XML format.

59. (Rejected and On Appeal) The method of claim 48, wherein the input files are in HTML format.

60. (Rejected and On Appeal) The method of claim 48, wherein the input files are in cHTML format.

61. (Rejected and On Appeal) The method of claim 48, wherein the input files are in WML format.

62. (Rejected and On Appeal) The method of claim 49, wherein the modified input files are compiled into DOM objects at runtime.

63. (Rejected and On Appeal) The method of claim 48, wherein the application framework code comprises an application object and a servlet web application framework object.

64. (Rejected and On Appeal) A method of generating computer code for a web application, comprising:

receiving input files, wherein the input files are at least one web application graphical user interface;

retrieving an application framework code from an application directory;

generating an event handler skeleton;

receiving web application business logic objects;

receiving event handler methods;

organizing the application framework code, the web application business logic objects and the event handler methods into web application source code;

and

binding modified input files with the web application source code at runtime and wherein generating an event handler skeleton further comprises:

parsing at least one input file;

reviewing the parsed input file for a tag type, an attribute name and an attribute value; and determining an event handler method based on the tag type, the attribute name and the attribute value.

65. (Canceled)

66. (Rejected and on Appeal) The method of claim 64, further comprising determining if the application framework code is available for the web application.

67. (Rejected and on Appeal) The method of claim 64, further comprising generating a business logic foundation code.

68. (Rejected and On Appeal) The method of claim 64, further comprising retrieving a business logic foundation code.

69. (Rejected and On Appeal) The method of claim 64, further comprising generating a graphical user interface code.

70. (Rejected and On Appeal) The method of claim 64, wherein generating an event handler skeleton is based on the input files.

71. (Rejected and On Appeal) The method of claim 64, wherein the input files are in XML format.

72. (Rejected and On Appeal) The method of claim 64, wherein the input files are in HTML format.

73. (Rejected and On Appeal) The method of claim 64, wherein the input files are in HTML format.

74. (Rejected and On Appeal) The method of claim 64, wherein the input files are in WML format.

75. (Rejected and On Appeal) The method of claim 64, further comprising compiling the modified input files at runtime.

76. (Rejected and On Appeal) The method of claim 64, further comprising interpreting the modified input files at runtime.

77. (Rejected and On Appeal) The method of claim 64, wherein the application framework code comprises an application object and a servlet web application framework object.

78. (Rejected and On Appeal) A method of generating computer code for a web application, comprising:

generating an event handler skeleton wherein the generating comprises parsing at least one input file, reviewing the parsed input file for one or more of a tag type, an attribute name and an attribute value, and determining an event handler method based on the one or more of tag type, the attribute name and the attribute value;

receiving a business logic foundation code, the event handler skeleton and a graphical user interface code; and

preparing web application business logic objects based on the business logic foundation code.

Claims 79-161 (canceled)

162. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

generate a business logic foundation code, an event handler skeleton and a graphical user interface code;

receive web application business logic objects from a web developer;

receive event handler methods from the web developer;

organize the application framework code, the web application business logic objects and the event handler methods into web application source code;

compile the web application source code;

compile the modified input files at runtime; and

bind the compiled modified input files with the compiled web application source code at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

163. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

generate an application framework code and an event handler skeleton; receive web application business logic objects; receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into application source code; and

bind the web application source code with the input files at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

164. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

retrieve an application framework code from an application directory;

generate an event handler skeleton;

receive web application business logic objects;

receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into application source code; and

bind the web application source code with the input files at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

165. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

generate an application framework code and an event handler skeleton; receive web application business logic objects; receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into web application source code; receive modified input files; and bind the modified input files with the web application source code at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

166. (Rejected and On Appeal) A device for generating computer code for a web application, comprising:

a storage device; and

a processor connected to the storage device, the storage device storing a program for controlling the processor;

the processor operative with the program to:

receive input files, wherein the input files are at least one web application graphical user interface;

retrieve an application framework code from an application directory;

generate an event handler skeleton;

receive web application business logic objects;

receive event handler methods;

organize the application framework code, the web application business logic objects and the event handler methods into web application source code;

receive modified input files; and

bind the modified input files with the web application source code at runtime and wherein the processor is operative with the program in generating the event handler skeleton to:

parse at least one input file;

review the parsed input file for one or more of a tag type, an attribute name and an attribute value;

and

determine an event handler method based on the one or more of a tag type, the attribute name and the attribute value.

167. (Rejected and On Appeal) The method of claim 2, further comprising:

determining if an application framework code is available for the web application; and if the application framework code is not available, then generating the application framework code.

168. (Rejected and On Appeal) The device of claim 166, further comprising:

a determining mechanism configured to determine if an application framework code is available for the web application; and

a code generator configured to generate the application framework code.

169. (Rejected and On Appeal) The device of claim 162, wherein the processor is further operative with the program to:

determine if an application framework code is available for the web application; and if the application framework code is not available, then generate the application framework code.

170. (Canceled)

171. (Canceled)

172. (Canceled)

173. (Canceled)

174. (Canceled)

X. EVIDENCE APPENDIX

There is no additional evidence to be submitted in connection with this Appeal Brief.

XI. RELATED PROCEEDINGS APPENDIX

No related Appeals are pending.